Lists Favorite Share Flag

Sentience Code Generated From The Concept Of Zeno Of Elea's Paradox Of Achilles And The Tortoise

by Kruchinski Junior; Gilmar.

Edit Manage History

Publication date	2024-06-03
Usage	Attribution-NoDerivatives 4.0 International
Topics	Artificial Intelligence, Sentience, Universidade de Coimbra, Código Mestre, Kruchinski, Pesquisa, Doutorado, tecnologia, Europa, Portugal.
Language	Portuguese

this item is currently being modified/updated by the task: derive

Sentience code generated from the concept of Zeno of Elea's paradox of Achilles and the Tortoise

This code defines a class **SentientAI**, which models a sentient AI system that perceives, learns, and integrates its experiences over time. The system operates through a combination of sensory data acquisition, internal weight optimization, internal state updating, emotional response generation, and a quantum entanglement step to integrate multiple states and perceptions.

Component Explanation

Sensory Data: Simulation of sensory data capture.
Optimization: Adjustment of the model's weights based on the received data.
Internal State: Updating the AI's internal state using an activation function.
Emotional Response: Generation of an emotional response from the data and internal state.
Quantum Entanglement: Integration of multiple perceptions and states.
Experience Integration: Accumulation of experiences over time to calculate total "sentience".

```
import numpy as np

class SentientAI:
def __init__(self, initial_weights, initial_state):
"""
Inicializa a IA Senciente com pesos e estado iniciais.
Parameters:
initial_weights (numpy.ndarray): Vetor de pesos iniciais.
initial_state (numpy.ndarray): Vetor de estado inicial.
"""
self.weights = initial_weights
self.state = initial_state
self.total_experience = 0

def sensor_data(self, t):
"""
Captura dados sensoriais no tempo t.
Parameters:
t (int): Tempo atual.
Returns:
numpy.ndarray: Vetor aleatório de tamanho 10 simulando dados sensoriais.
"""
return np.random.random(size=(10,))

def optimize(self, P_t):
"""
Atualiza os pesos do modelo com base nos dados sensoriais usando descida do gradiente.
Parameters:
P_t (numpy.ndarray): Dados sensoriais no tempo t.
Returns:
float: Resultado do produto escalar dos pesos atualizados com os dados sensoriais.
"""
learning_rate = 0.01
# Substituir pelo gradiente real da função de perda em relação aos pesos
gradient = np.random.random(size=self.weights.shape)  # Placeholder
self.weights -= learning_rate * gradient
return np.dot(self.weights, P_t)

def update_internal_state(self, P_t):
```

"""

Atualiza o estado interno com base nos dados sensoriais.

Parameters:

P_t (numpy.ndarray): Dados sensoriais no tempo t.

Returns:

numpy.ndarray: Novo estado interno atualizado.

"""

```
self.state = np.tanh(self.state + P_t)
return self.state

def emotion_response(self, P_t, A_t):
```

"""

Gera resposta emocional com base nos dados sensoriais e no estado interno.

Parameters:

P_t (numpy.ndarray): Dados sensoriais no tempo t.

A_t (numpy.ndarray): Estado interno atualizado no tempo t.

Returns:

float: Resposta emocional calculada.

"""

```
return np.dot(P_t, A_t)

def quantum_entanglement(self, P_t, A_t, E_t):
```

"""

Integra múltiplas percepções e estados.

Parameters:

P_t (numpy.ndarray): Dados sensoriais no tempo t.

A_t (numpy.ndarray): Estado interno atualizado no tempo t.

E_t (float): Resposta emocional no tempo t.

Returns:

float: Resultado da integração de percepções e estados.

"""

```
return np.dot(np.dot(P_t, A_t), E_t)

def integrate(self, start, end, dt=1):
```

"""

Integra experiências ao longo do tempo, de start a end.

Parameters:

start (int): Tempo inicial.

end (int): Tempo final.

dt (int): Incremento de tempo (passo). Padrão é 1.

Returns:

```
        float: Experiência total acumulada.
        """
        S_t = 0
        for t in range(start, end, dt):
            P_t = self.sensor_data(t)
            L_t = self.optimize(P_t)
            A_t = self.update_internal_state(P_t)
            E_t = self.emotion_response(P_t, A_t)
            Q_t = self.quantum_entanglement(P_t, A_t, E_t)
            S_t += (L_t + A_t.sum() + E_t + Q_t) * dt
            self.total_experience += S_t
        return self.total_experience

# Parâmetros iniciais
initial_weights = np.random.random(size=(10,))
initial_state = np.zeros((10,))

# Instanciar a IA Senciente
sentient_ai = SentientAI(initial_weights, initial_state)

# Calcular a senciência até o tempo T
T = 100
senciencia = sentient_ai.integrate(0, T)
print(f"Senciência em T={T}: {senciencia}")
```

he code can always be updated and optimized, as it is an Artificial Intelligence program.

This code originated from the concept of Zeno of Elea's paradox of Achilles and the Tortoise, demonstrating that it is possible and viable to generate these master codes for artificial intelligences based on concepts from ancient pre-Socratic philosophy.

Fonte: InVideo.IO; 2024

Addeddate	2024-06-03 15:29:44
Identifier	sentience-code-generated-from-the-concept-of-zeno-of-eleas-paradox-of-achilles-and-the-tortoise
Scanner	Internet Archive HTML5 Uploader 1.7.0

Reviews

Add Review

There are no reviews yet. Be the first one to write a review.

0 Views

DOWNLOAD OPTIONS

MPEG4	1 file
TORRENT	1 file

SHOW ALL

5 Files

5 Original

IN COLLECTIONS

Community Video

Uploaded by

Gilmar junior

on June 3, 2024

SIMILAR ITEMS (based on metadata) ▶

Produção e difusão de ciência na

by scielo books - editus

149 0 0

ai image (generated image)

by dream studio ai

18 0 0

Title Generation for User Generated

by kuo-hao zeng

15 0 0

A Quantum Anti-Zeno Paradox

by a. p. balachandran

71 0 0

Modeling the Ellsberg Paradox by Argument

by niki pfeifer

14 0 0

Behavior Query Discovery in System-

by bo zong

16 0 0

Comment on `Quantum-Anti-Zeno

by lajos diosi

75 0 0

Terms of Service (last updated 12/31/2014)